Dog Image Generation using Deep Convolutional Generative Adversarial Networks

Zhongkai Shangguan
Department of Electrical and Computer Engineering
University of Rochester
Rochester, US
zshangg2@ur.rochester.edu

Yue Zhao
Department of Electrical and Computer Engineering
University of Rochester
Rochester, US
yzhao88@ur.rochester.edu

Wei Fan
Department of Electrical Engineering
Columbia University
New York, US
wf2271@columbia.edu

Zhehan Cao
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, US
zcao76@gatech.edu

2020 5th International Conference on Universal Village (UV) | 978-1-7281-9523-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/UV50937.2020.9426213

***Abstract*—Generative Adversarial Networks (GAN) serve as an important position of the data generation models, providing possibility for generating nonexistent images, style transfer, background masking, alternative faces, etc. However, the generated images are becoming more and more realistic, which has raised the concern of people's privacy. In this paper, we implemented a Deep Convolutional Generative Adversarial Network (DCGAN) to show how to generate novel dog images from noise. We improved the performance of the basic DCGAN by applying different tricks, including adding noise to the training images, excute input normalization and batch normalization, comparing different activation functions, and using soft labels. The purpose of all these tricks is to synchronize the learning process between generator and discriminator as well as introduce stochasticity. The performance evaluation is based on Memorization-informed Frechet Inception Distance (MiFID) and results show that the MiFID value of our model reached outstanding performance, which is 95.85.**

Keywords*—Generative Adversarial Networks, privacy, Deep Convolutional Generative Adversarial Network, Memorization-informed Frechet Inception Distance

I. Introduction and Related Work

With the rapid development of artificial intelligence, many popular applications have been developed, e.g., background masking [1], color palette completion [2], parameter optimization for power grid system [3], interactive anime [4], etc. Among them, Generative Adversarial Net (GAN) [5] is a very popular technique that have great potential in simulating data distributions and make computers creative.

Past few years witnessed the rapid development of GAN, and showed that GAN have achieved great success at generating realistic and sharp looking images. There are two main components of GAN: a generator that used to create the fake images and a discriminator which is used to identify whether the images are from the dataset or generated. There are a series of different models of GAN such as Conditional Generative Adversarial Nets (CGAN) [6], Deep Convolutional GAN (DCGAN) [7], Cycle-Consistent Adversarial Networks (CycleGAN) [8], etc.

Inspired by exploring more intriguing applications, we focus on generating images of dogs that never existed before by designing a symmetric DCGAN. DCGAN is an extension of the GAN architecture: instead of using fully-connected layers, it applies convolutional layers [9] in the discriminator to extract the feature map, and uses convolutional-transpose layers in the generator. In this paper, we aim to improve the quality of the generated dog images and make them more realistic looking. In order to achieve our goal, various techniques are applied during train process, such as tuning hyper-parameters, adding noise to the input images, using soft and noisy label, etc. We use the Memorization-informed Frechet Inception Distance (MiFID) [10] to evaluate our performance.

Such technology that can be used to generate fake images can also be used for malicious purposes, which raise people's concern of their privacy, and many companies have to research new technologies to identify fake content [11]. However, on the other hand, GAN can also be used to protect people's privacy. Wu et al. proposed a Privacy Protective GAN that could generate a identity of a face to do face de-identification [12]; with this technology, people who have to supply their personal face information could submit the identity directly, instead of sharing their real face information. In this case, applications could be developed to block out the information that people do not want to share.

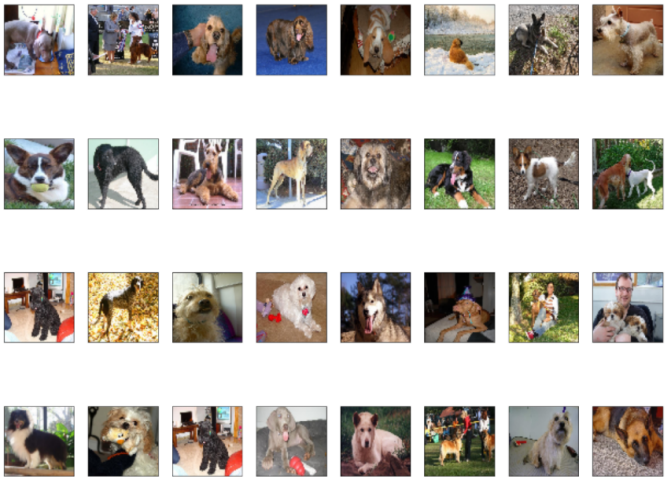

Fig. 1. Image samples before data cleaning

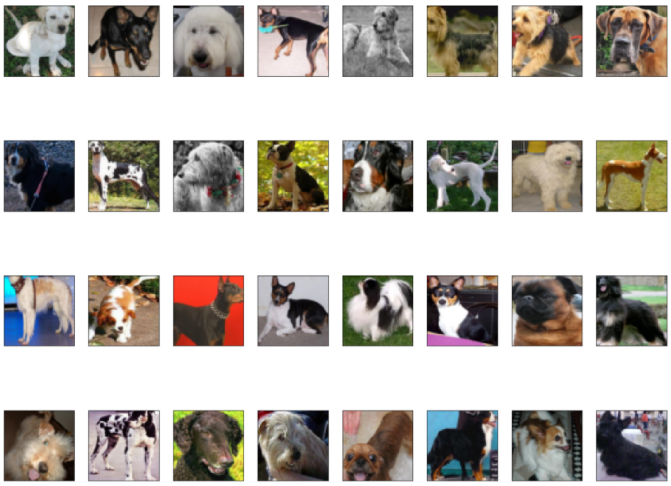

Fig. 2. Image samples after data cleaning

The rest of the paper is arranged as follows. The dataset, data pre-processing methods and the evaluation metrics are described in Section Dataset. We then introduce our model structure and the tricks we applied during the train process in Section Methodology. Section Results shows the experiment results of our model. We finally make conclusions and future work.

II. Dataset and Evaluation Metrics

In this section, we describe the dataset we use and how we process the data; then we show how we assess the performance by introducing the evaluation metrics.

A. Dataset and Data-preprocessing

The dataset we use is Stanford Dogs dataset [13]. The dataset contains 20580 dog images, with annotations of class labels and bounding boxes, i.e., which breed is the dog and its precise location coordinates in the images. Most areas of the images are dogs, however, different images have different perspective to the dogs, and the images also contain background noises such as people, nature scenes and objects, as shown in Fig. 1.

In our project, we regard the dogs all of the same category and do not distinguish the breeds of them. In order to provide data consistency, we perform data cleaning for the dataset, including cropping operation based on the bounding boxes annotations and discarding low-quality images. More specifically, the images do not maintain a $1:1$ aspect ratio after the crop operation, so we need to resize the images before they are fed to the neural networks since neural networks require a certain size input. However, the image geometric transformation will cause loss of information or distortion of the features, especially for images with large aspect ratio. Under this condition, we performed a further filter for the input data. In our case, we drop images that aspect ratio is larger than 1.25 or smaller than 0.8 after cropping, and resize them to the shape of $(64, 64)$. After the data cleaning process, we obtained a subset of the Stanford Dogs dataset with 9070 images that are used as training data, and Fig. 2 shows some samples of the cleaned dataset. We further applied data augmentation stage including "RandomResizedCrop" and "RandomHorizontalFlip" [14].

B. Evaluation Metrics

To evaluate the performance of our model, we use the Memorization-informed Frechet Inception Distance (MiFID) [10], which is modified from Frechet Inception Distance (FID) [15], to measure the distance between images from the dataset and our generations.

The FID score uses the *inception v3* model [16]. Specifically, the feature extractor of the *inception v3* model is used to extract the feature map of the input image [17]. Then we introduce a multivariate Gaussian distribution. The mean μ and covariance Σ of the distribution which we introduced are used to model the data distribution for the features extracted from an the *inception v3* network across both real and generated images. The distance between these two distributions is then calculated using the Frechet distance [18].

The mathematic equation for the FID is:

$$FID = \|\mu_r - \mu_s\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}\right) \quad (1)$$

where Tr represents the trace; r is the real images and the generated images is g; μ_r, Σ_r and μ_g, Σ_g are the mean value and covariance of the multivariate Gaussian distribution for real and generated images respectively.

However, one drawback of FID is that it does not consider the overfitting situation, e.g., if the images generated are quietly similar to the input dataset, as the result, the FID must be very small, but at this time the network only remembers the training dataset instead of creating the images. In addition to FID, Kaggle [10] use MiFID to avoid overfit. The memorization distance is defined as shown in (2).

$$d_{i,j} = 1 - cos(f_{gi}, f_{rj}) = 1 - \frac{f_{gi} \cdot f_{rj}}{|f_{gi}||f_{rj}|} \quad (2)$$

where f_g is the feature space of the generated data; f_r is the feature map of the data collection from the real world; f_{gi} and f_{rj} are the i^th and j^th vectors of f_g and f_r, respectively.

Then we take a certain generated image (i), and find the minimum distance between it and all images (j). Finally we take the averaged, as shown in (3).

$$d = \frac{1}{N} \sum_{i} \min_{j} d_{i,j} \tag{3}$$

where N represent the number of generated images.

This distance d is thresholded using a pre-defined epsilon ϵ, and it will be one if the distance exceeds a value we designed before, i.e.,

$$d_{thr} = \begin{cases} d, & if\ d < \epsilon \\ 1, & otherwise \end{cases} \tag{4}$$

and finally, MiFID is defined as

$$MiFID = FID \cdot \frac{1}{d_{thr}} \tag{5}$$

where d_{thr} is the memorization distance threshold. Same to FID, A smaller MiFID value means that a better image was generated [10]. The overall evaluation process is shown in Fig. 3.

III. Methodology

In this section, we firstly demonstrate the loss function and the model architecture of our DCGAN, then describe the training methods we use in detail.

A. Model Architecture

Generative Adversarial Network (GAN) [19] is widely used in generating images that never existed in real world. It contains two essential components: a generator and a discriminator. The technology works by learning the distributions of a specific type of data and creating new data of the same type from a so-called latent space, which is noisy. Fig. 4 shows the structure of GAN. The input of the generator is noise, also known as the latent space, and the generator will take these noise to generate fake dog images. Along with real-world images, these fake images will be fed into the discriminator, which is a binary classification model that tries to classify fake data from real images. Finally, the output of the discriminator will be used to calculate the loss. The loss we used is Binary Cross-Entropy (BCE) [20], defined by:

$$BCE = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log\left[p\left(y_i\right)\right] + \left(1 - y_i\right) \cdot \log\left[1 - p\left(y_i\right)\right] \tag{6}$$

where N is the number of images that are trained in one batch, y_i, and $p(y_i)$ are the ground truth and predicted values for i^{th} image.

The entire process is a minimax problem where generator and discriminator play against each other. The goal of discriminator is to perfectly distinguish which images are fake ones and which are real, i.e., to minimize the loss function; while the goal of generator is to make the generated images

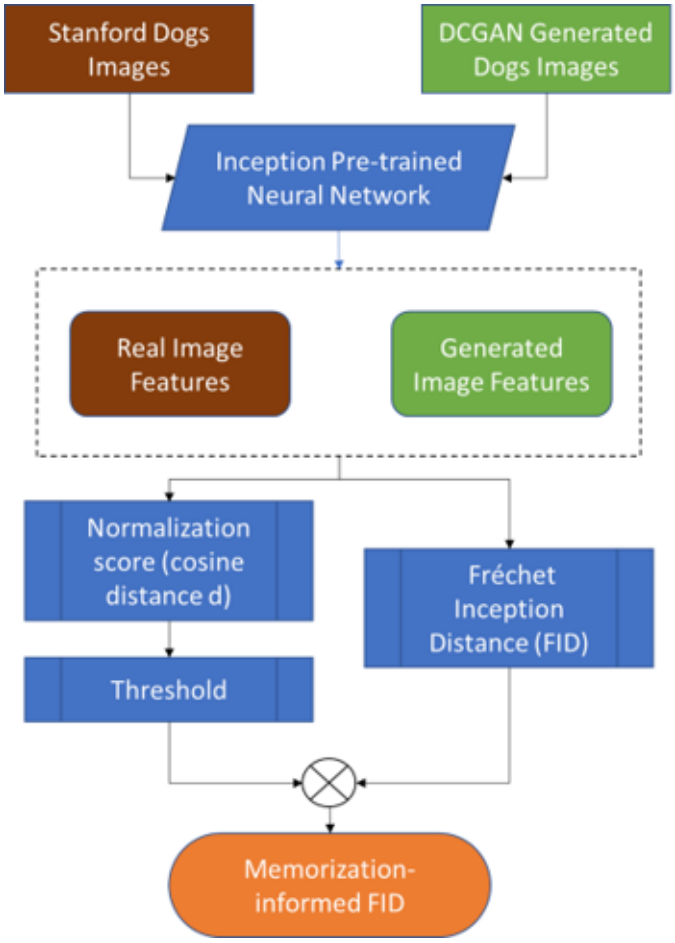

Fig. 3. Image samples after data cleaning

more like the real images, so that the discriminator cannot distinguish which are the fake images, which will increase the loss function. Substitute discriminator and generator into (6), then we optimize the following equation:

$$\begin{aligned} \min_{D} \max_{G} L(D, G) &= BCE(x) + BCE(G(z)) \\ &= -\frac{1}{N} \sum_{i=1}^{N} logD(x) + log(1 - D(G(z))) \end{aligned} \tag{7}$$

where x and $G(z)$ represent the real and fake images, $D(x)$ represents the probability that x is from the real dataset while $D(G(z))$ is the probability that the discriminator regards the images generated by generator are real.

The discriminator and the generator are trained alternately, i.e., training the discriminator first, then training the generator, and reciprocating continuously. Specifically, the generator will generate images and feed them to the discriminator together with the real data, then update the parameters in the discriminator. Then use the updated discriminator to recalculate the loss function, and update the trainable parameters in the generator. In the next epoch of training, use the updated generator to regenerate the images and repeat the above steps. In the whole process, the generator is trying to generate more

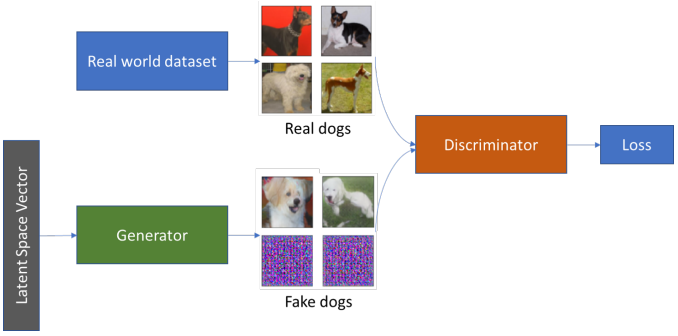

Fig. 4. Structure of GAN

real images, and at the mean time, the discriminator is striving to classify which images are fakes ones. The generator and the discriminator are competing each other, and finally the two networks will reach a balanced point, i.e., the generated image by the generator is very like the real image, and the discriminator cannot find the real and fake images.

A DCGAN is a direct extension of the GAN [9]. As shown in Fig. 5, the discriminator is comprised of convolution layers and the generator is comprised of convolutional-transpose layers.

A very important issue of DCGAN is how to balance the learning capabilities of the generator and the discriminator. As the main function of the discriminator is to provide a descending gradient for the generator, so if the discriminator is too bad, it cannot provide a valid gradient, while if we train the discriminator too good, the gradient disappears. Therefore, the discriminator and generator need to compete on the same level, that is, the ability of the discriminator cannot be much better or worse than the generator. To resolve the above issue, we develop a symmetric architecture for generator and discriminator, as shown in Fig. 5. For convolutional neural networks, it is common known that the deeper the learning ability. Therefore, we balance the learning ability between generator and discriminator by letting them have the same depth of the convolutional neural networks, as well as the size of the convolutional kernel.

B. Training Methods

GAN is hard to be trained as decribed in [21] :

- Non-convergence: it is hard to make the generator and discriminator converge at the same time. Simultaneous gradient descent for both models makes them converge but not both of them reach the optimal convergence at the same time, which may make one model (e.g., discriminator) much stronger/weaker than the other (e.g., generator).
- Mode collapse: the generator makes different images contain the same features such as the color, or different images contain different parts of the same dog.
- Diminished gradient: if the discriminator gets too successful, then the gradient of the generator is hard to be updated, therefore the generator learns nothing.

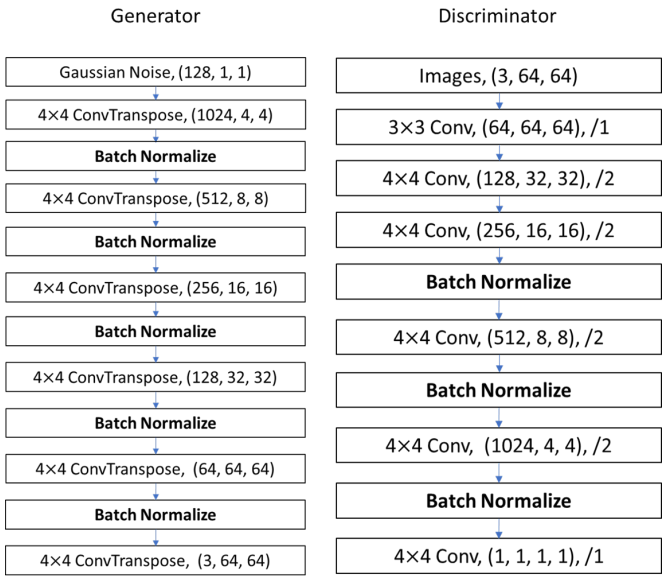

Fig. 5. Generator and Discriminator Architecture of DCGAN

In order to train our DCGAN stably and effectively, exquisite training methods need to be designed. In our project, in order to balance the performance and the learning progress as well as make the training process much smoother, following techniques are applied.

1) Input Normalization: The values of all input images are normalized between -1 and 1 , and we use the hyperbolic tangent (Tanh) [22] the last layer of the generator so that the value will between -1 to 1, which is consistent with the interval range of the original image.

2) Add Noise to Input Images: Arjovsky et al. [23] proposed that adding noise to input images and decay during the training process could help. In our experiment, we add Gaussian noise with $mean = 0$ and $variance = 0.01$ to the input image, as experience shows the discriminator is easier to be train compared with the generator.

3) Data Augmentation: Data augmentation is considered as a basic and simple method that direct to the improvement of model performance. We applied RandomResizedCrop [14] to crop the given images to $(0.8, 1.0)$ of the original size and a random aspect ratio between $(3/4 to 4/3)$ of the original aspect ratio, then resize the images to $(64, 64)$. RandomHorizontalFlip [14] is also applied.

4) Batch Normalization: Batch normalization (BN) [24] is also used to construct different mini-batches for real and generated images. This means each mini-batch should only include either a pure real image or a pure generated image. The idea of BN is to normalize the input of the current layer, making the mean input 0 and the variance 1. The advantage of using BN is to accelerate the convergence process, and the convolutional neural network added by BN is not significantly affected by the weight initialization, which increase its stability and robustness, and has a good effect on improving the convolution performance.

TABLE I
HYPER-PARAMETERS SETTING

learning rate (generator & discriminator)	0.0005
betas (parameter of Adam [26])	(0.5, 0.999)
batch size	32
length of latent space	128
epochs	750

5) Avoid Sparse Gradient: The stability of DCGAN will be greatly affected by the introduction of sparse gradients. Therefore, we use convolutional layer with stride to perform downsample and onvolutional-transpose layer with stride to implement upsample. LeakyReLU is also used in both the generator and the discriminator. By doing such operation, this will enable the process of training is smooth, which helps balance the learning progress between the generator and the discriminator.

6) Use soft label: The soft and noisy labels are used to smooth the training process. For the label smoothing, instead of setting the target labels as $real = 1$ and $fake = 0$, we label the incoming real sample with 0.8. However, for each incoming sample of fake images, we still keep the label $fake = 0$.

7) Adam Optimizer: Adam optimizer is regarded as a efficient and effective optimizer for the training process [25].It is very popular in the field of deep learning because it can achieve excellent results quickly.

IV. EXPERIMENTS AND RESULTS

We train 750 epochs with the techniques applied as discussed in section III. Other settings of hyper-parameters are shown in Table I. The input data for the generator is a 128 dimension random Gaussian noise. The code is available at https://www.kaggle.com/leonshangguan/dcgan-data-cleaning-sub-v1?scriptVersionId=18470145.

Fig. 7 demonstrates the validation $MiFID$ during the training process, and we finally achieved $MiFID = 95.85$ in the test dataset and 38.70 in the validation dataset. Fig. 6 shows the generated dog images using our DCGAN. As it can be seen, it is difficult to distinguish whether they are generated fake dogs or real dogs, which indicates a good performance.

V. CONCLUSION AND FUTURE WORK

This paper illustrates the modified model of Generative Adversarial Nets (GAN) to generate dog images that are never created before. The modified symmetric Deep Convolutional GAN (DCGAN) architecture is used and various tricks during training process are applied in order to balanced the learning ability between the generator and the discriminator. The final $MiFID$ of our model reaches 95.85 in the testing phase and 38.70 in the validation phase, which indicates a good performance so that the images generated are likely to be detected as real images.

We consider three aspects in our future work: i) In order to make comparisons, other GAN structures are worth to try, e.g., Cycle-Consistent Adversarial Networks (CycleGAN). ii)

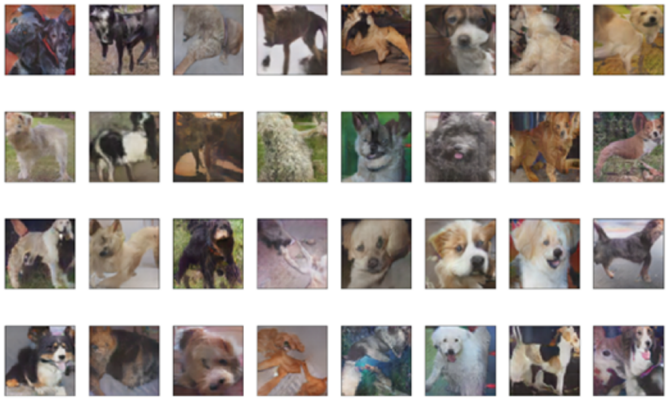

Fig. 6. Generated Dog Images using DCGAN

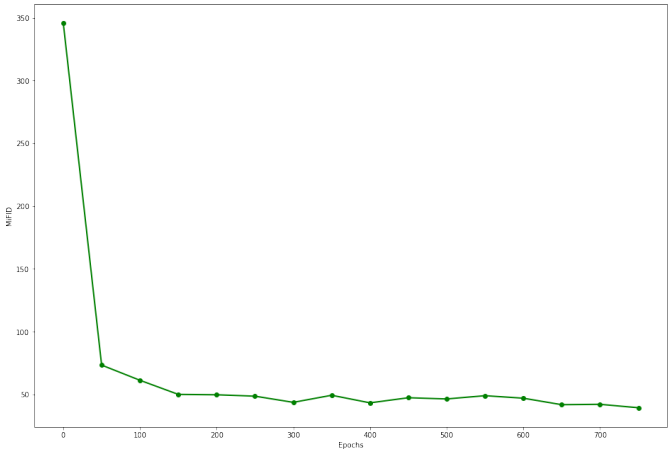

Fig. 7. Validation MiFID during Training Process

As the application of generate images also create privacy concerns, more specific, complete, and detailed laws and policies are needed, and this is a significant ethical concern for the Universal Village [27]. iii) The technology can also be used to protect people's privacy as described in [12]. For example, the online dating applications probably can help chatters who do not want to disclose personal face information to make a fake face that still contain chatters' facial characters, which could protect the chatter's privacy, but still make an intuitive impression between the chatters.

REFERENCES

[1] S. Gu, J. Bao, H. Yang, D. Chen, F. Wen, and L. Yuan, "Mask-guided portrait editing with conditional gans," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3436–3445, 2019.

[2] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu, "Unsupervised diverse colorization via generative adversarial networks," in Joint European conference on machine learning and knowledge discovery in databases, pp. 151–166, Springer, 2017.

[3] L. Lin, W. Wu, Z. Shangguan, S. Wshah, R. Elmoudi, and B. Xu, "HP-TRL: Calibrating power system models based on hierarchical parameter tuning and reinforcement learning," in 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2020.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134, 2017.
[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in neural information processing systems, vol. 27, pp. 2672–2680, 2014.
[6] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.
[7] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.
[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in Proceedings of the IEEE international conference on computer vision, pp. 2223–2232, 2017.
[9] N. Inkawhich, "Dcgan tutorial," 2017. [Online]. Available: https: //pytorch.org/tutorials/beginner/dcgan faces tutorial.html.
[10] K. Team, "Generative dog images - experiment with creating puppy pics — kaggle.," 2020. [Online]. Available: https://www.kaggle.com/c/generative-dog-images/overview/evaluation.
[11] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," arXiv preprint arXiv:2001.00179, 2020.
[12] Y. Wu, F. Yang, Y. Xu, and H. Ling, "Privacy-protective-gan for privacy preserving face de-identification," Journal of Computer Science and Technology, vol. 34, no. 1, pp. 47–60, 2019.
[13] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition, (Colorado Springs, CO), June 2011.
[14] TorchContributors., "Torchvision.transforms," 2020. [Online]. Available: https://pytorch.org/docs/stable/torchvision/transforms.html.
[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," Advances in Neural Information Processing Systems, vol. 30, pp. 6626–6637, 2017.
[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826, 2016.
[17] J. Brownlee, "How to implement the frechet inception distance (fid) for evaluating gans," 2019. [Online]. Available: https://machinelearningmastery.com/ how-to-implement-the-frechet-inception-distance-fid-from-scratch/.
[18] D. Dowson and B. Landau, "The fr´echet distance between multivariate normal distributions," Journal of multivariate analysis, vol. 12, no. 3, pp. 450–455, 1982.
[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," Communications of the ACM, vol. 63, no. 11, pp. 139–144, 2020.
[20] J. Brownlee, "A gentle introduction to cross-entropy for machine learning," 2019. [Online]. Available: https://machinelearningmastery.com/cross-entropy-for-machine-learning/.
[21] Medium, "Gan — why it is so hard to train generative adversarial networks.," 2020. [Online]. Available: https://medium.com/@jonathan hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks.
[22] WikipediaContributors., "Hyperbolic functions," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Hyperbolic functions.
[23] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," arXiv preprint arXiv:1701.04862, 2017.
[24] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in Advances in neural information processing systems, pp. 2483–2493, 2018.
[25] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to train a gan? tips and tricks to make gans work," 2016. [Online]. Available: https://github.com/soumith/ganhacks.
[26] TorchContributors., "Torch.optim," 2020. [Online]. Available: https://pytorch.org/docs/stable/optim.html.
[27] Y. Fang, B. Horn, and I. Masaki, "Universal village: What? why? how?," in The First International Conference on Universal Village, 2013.